Microsoft .NET, conviene davvero?

In questo articolo forniremo una panoramica sulla nuova tecnologia Microsoft per gli sviluppatori e cercheremo di capire le difficoltà che i "vecchi" programmatori devono affrontare per trarre vantaggio dai nuovi strumenti.

Lorenzo Barbieri è leader della Technology Practice Microsoft .NET in Object Way (www.objectway.it)

È passato un anno dal rilascio ufficiale della versione 1.0 del Microsoft .NET Framework e del Visual Studio.NET, e molto è stato detto e scritto in questi mesi a proposito del nuovo ambiente di sviluppo, e dell'impostazione che bisogna seguire per sviluppare applicazioni per la nuova piattaforma. Eppure ci sono ancora molte realtà che sviluppano in ambiente Microsoft a cui gioverebbe molto passare a .NET ma che per vari motivi non l'hanno ancora fatto.
I motivi principali che ho sentito in questi mesi erano:

- non usiamo una tecnologia appena rilasciata;
- ma cos'è questo .NET, non si capisce bene;
- passare a .NET ci costerebbe troppo e non porterebbe nessun vantaggio;
- non voglio più legarmi mani e piedi ad una tecnologia proprietaria, il mio software deve essere aperto;

Per quanto riguarda il primo motivo, in parte si può pensare di dare ragione, ma bisogna considerare che sono già usciti due service pack e che il Framework è stato testato considerevolmente prima di raggiungere la versione finale. Vediamo invece di capire in breve cos'è .NET e i vantaggi che porterebbe il suo utilizzo.

COS'È MICROSOFT .NET

.NET è la visione Microsoft per lo sviluppo di applicazioni e soluzioni basate sui servizi.
C'è stata molta confusione a riguardo, dovuta in parte ad alcune scelte 'discutibili' del marketing Microsoft, come quella di targare come .NET Servers alcuni prodotti (come SQL Server 2000, Exchange 2000, etc...) che con il .NET Framework non avevano nulla a che fare, o quella più generale di mettere la sigla .NET all'interno di quasi tutti i propri prodotti. Adesso la visione è un po' più chiara e si delinea come:

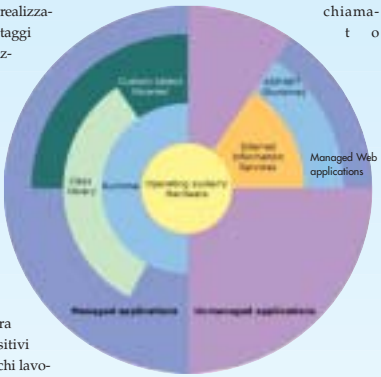
- **una piattaforma di sviluppo** - il .NET Framework
- **i server su cui gireranno le applicazioni** - Windows 2000 Server o meglio il nuovo Windows .NET Server 2003
- **i server di supporto** - i 'vecchi' .NET Servers che possono interagire con il Framework come SQL Server 2000 o i nuovi Commerce Server 2002 e BizTalk Server 2002 che hanno nativamente il supporto del Framework

- **i client che faranno girare le applicazioni .NET** – i PC Windows, i browser, i dispositivi basati su Windows CE e Windows CE.NET, i telefonini internet enabled, etc...
- **una serie di servizi forniti da Microsoft e/o da terze parti** – ad esempio .NET Pass Port, MapPoint.NET, .NET Alerts, etc...
- **un ambiente di sviluppo con cui programmare e coordinare tutti questi strumenti** – Visual Studio.NET.

L'utilizzo dei server, dei client e dei servizi è una scelta che dipende dal tipo di applicazione che si intende sviluppare, mentre la scelta se utilizzare o no il Framework è indipendente da quello che si intende realizzare. Cerchiamo quindi di vedere i vantaggi dell'utilizzo del Framework per realizzare le proprie applicazioni.

.NET FRAMEWORK, FUNZIONALITÀ E VANTAGGI

Il .NET Framework è la piattaforma applicativa con cui è possibile sviluppare in maniera omogenea e coerente applicazioni desktop, client/server, applicazioni internet enabled, applicazioni intra /extra /internet, applicazioni per dispositivi smart e mobili, etc... Il vantaggio per chi lavora in ambiente Microsoft è che utilizzando il Framework si unificano mondi che prima erano profondamente separati, come VB6 per la realizzazione di applicazioni desktop, ASP/HTML/ JavaScript per le applicazioni Web, Visual C++ per la realizzazione di applicazioni che dovevano essere performanti, scalabili, etc... Ognuno di questi ambienti aveva un modello di programmazione differente, classi base differenti, filosofie differenti. Con il Framework tutto questo non è più vero. Vediamo quindi come è stato possibile ottenere questa unificazione.



COMMON LANGUAGE RUNTIME

Per unificare i vari modelli di programmazione è stato creato un ambiente di esecuzione per le applicazioni chiamato Common Language Runtime che si occupa di gestire la memoria, di eliminare gli oggetti e le strutture dati non più necessarie (Garbage Collection), di gestire il multithreading, di gestire il debugging locale e remoto, di verificare la provenienza del codice ed i suoi permessi di esecuzione del codice, etc...

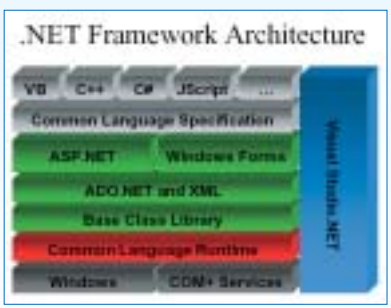
Questo ambiente di esecuzione non interagisce con il codice nativo (x86 nel caso di macchine Windows) generato dai compilatori tradizionali ma richiede che il codice venga compilato in un linguaggio chiamato Microsoft Intermediate Language (MSIL) che verrà poi compilato Just-In-Time nel codice nativo del processore da parte del CLR.

Il fatto che il codice venga compilato direttamente sulla macchina che lo eseguirà porta lo svantaggio di un tempo di avvio maggiore (durante la prima compilazione) ma il codice generato sarà ottimizzato per il processore in uso (ad esempio codice P3 o P4 al posto di codice x86 generico). Per ovviare alla lentezza della prima compilazione è anche possibile effettuarla in fase di installazione del programma.

Il CLR porta enormi vantaggi nello sviluppo, come il fatto che il programmatore può disinteressarsi di molti aspetti di infrastruttura che vengono gestiti automaticamente dall'ambiente, senza la necessità di dover riscrivere centinaia di volte sempre lo stesso codice, con il rischio di introdurre errori. Il CLR poi fornisce i servizi in maniera uguale a tutti i linguaggi che dispongano di un compilatore che generi codice MSIL, quindi non ci sono più linguaggi di serie A e di serie B, ma solo linguaggi che sfruttano più o meno efficacemente le risorse che gli vengono messe a disposizione.

CLASSI DI BASE

Sopra il CLR sono disponibili centinaia di classi che costituiscono il cuore del Framework.. Ci sono classi per gestire gli oggetti di base, classi che forniscono accesso alle varie risorse del sistema operativo, alla rete, classi che permettono di manipolare dati relazionali o in formato XML, classi per realizzare applicazioni desktop (Windows Form), classi per realizzare applicazioni internet e Web Service (ASP.NET), etc... Le funzionalità delle classi di base sono disponibili per tutti i linguaggi compatibili con .NET, consentendo di scegliere il linguaggio che si preferisce, senza preoccuparsi di quali funziona-



lità si perdono o si guadagnano. Inoltre la struttura delle classi è completamente modulare ed accessibile, cosa che ne consente l'estensione ed il riutilizzo secondo una logica pienamente object-oriented.
Ad esempio è possibile realizzare form che possono essere ereditate da altre form, permettendo di scrivere il codice comune una volta sola, ed essere sicuri che ad ogni modifica tutte le classi figlie verranno aggiornate automaticamente.
Questo perchè tutte le funzionalità delle classi di base sono realizzare con del codice, e non sono dei componenti 'magici' forniti dal sistema, come ad esempio le form di VB6.

LINGUAGGI .NET E COMMON LANGUAGE SPECIFICATIONS

Il fatto che un linguaggio .NET generi codice MSIL che gira nell'ambiente di runtime comune non basta per garantire che il linguaggio possa sfruttare le caratteristiche delle classi di base, e non garantisce nemmeno che due differenti linguaggi possano interagire tra di loro. Per questo sono state create le Common Language Specifications (CLS) che forniscono un insieme di regole su come i vari linguaggi debbano gestire i tipi base, gli oggetti, le classi, etc...
Grazie a queste regole è possibile utilizzare in un programma scritto in Visual C# un componente scritto in Visual Basic.NET (cosa che era possibile anche con COM), ma è anche possibile estendere una classe o un componente scritto in un linguaggio diverso esattamente come se fosse stato scritto nello stesso linguaggio.
E' possibile (anche se sconsigliato) scrivere un'applicazione dove ogni classe è scritta in un linguaggio diverso. E' anche possibile durante il debugging saltare da un codice sorgente all'altro in maniera nativa, sempre all'interno del debugger.

QUANTO CI METTO A PASSARE A .NET

Dopo aver visto le caratteristiche principali del .NET Framework, e soprattutto i vantaggi che porta nello sviluppo delle applicazioni, la domanda che sorge più spesso è legata alla difficoltà del passaggio a .NET e al tempo necessario a portarlo a termine. Certamente tutto questo è legato all'esperienza del team di sviluppo, alle conoscenze acquisite, e alla flessibilità nell'apprendere nuovi modi di agire, ma dopo un'esperienza di quasi un anno si può dire tranquillamente che il passaggio a .NET del team di sviluppo è un'operazione abbastanza tranquilla.
Se un team di sviluppo conosce molto bene Visual C++, passare al Visual C++.NET o ancor meglio in certi casi al Visual C# è molto semplice, si tratta solo di capire le funzionalità che il CLR mette a disposizione e lasciar fare a lui (in maniera migliore) quello che prima si faceva a mano. Se un team è abituato a VB, può tranquillamente passare a Visual Basic.NET, dovendo solo imparare le piccole differenze tra i due linguaggi, e le nuove funzionalità che le classi base mettono a disposizione.
Gli sviluppatori ASP troveranno in ASP.NET un ambiente molto semplice da usare, e che fornisce già pronte centinaia di funzionalità che prima bisognava crearsi a mano, ad esempio il caching delle pagine, le griglie, etc...
Se un gruppo programmava in Java, passare a Visual C# o a Visual J# è immediato, con solo il dovere (nel primo caso) di imparare una nuova serie di classi base. Programmatori abituati ad altri linguaggi (PERL, Python, Fortran, COBOL, etc...) farebbero bene a guardare le implementazioni .NET dei loro linguaggi preferiti, e scoprire le nuove funzionalità che avranno a disposizione.

PORTING DELLE APPLICAZIONI

Se far conoscere al proprio team di sviluppo il linguaggio .NET più appropriato non è poi cosi lungo e impegnativo, cosa si può dire del porting di applicazioni già esistenti in .NET? La risposta più giusta è: dipende dai casi.

Se un'applicazione è stata sviluppata utilizzando componenti COM, con un'architettura a più livelli molto disaccoppiati la mossa migliore è di migrare piano piano i vari livelli, cominciando dai più semplici, sfruttando l'interoperabilità tra COM e .NET per far comunicare i componenti nuovi con quelli vecchi. Anche se l'applicazione è stata realizzata in VB utilizzando form e controlli ActiveX, è possibile migrare l'applicazione per fasi. In questo caso la scelta è se migrare prima l'infrastruttura dell'applicazione o i componenti, ma questo dipende fortemente dal singolo caso. Se l'applicazione è stata realizzata in Visual C++ è possibile ricompilarla con .NET semplicemente abilitando uno switch del compilatore. In questo modo è possibile iniziare a sfruttare le classi del Framework senza dover riscrivere l'applicazione.
La migrazione può essere fatta poi classe per classe se e quando se ne sente la necessità. Il consiglio per le applicazioni ASP è di passare il più velocemente possibile ad ASP.NET, considerando che sullo stesso Web Server è possibile far convivere i due ambienti, e quindi usare anche qui la strategia della migrazione incrementale, cercando però di portarla a termine nel più breve tempo possibile. Prima di iniziare a migrare tutte le applicazioni in .NET ponetevi però la domanda più importante: cosa ci guadagno? Se l'applicazione o il modulo va già bene cosi potete mantenerlo nel linguaggio originale, non affrettatevi ad effettuare la migrazione.
La migrazione potete affrontarla nel momento in cui dovete

aggiornarla, cambiarla o semplicemente nel momento in cui vi accorgete che l'originale non riesce più a reggere il carico. In alcuni casi la strategia della migrazione incrementale e/o parziale non è applicabile, e quindi bisogna riscrivere l'applicazione completamente da zero.

Questo è consigliato solo quando l'architettura dell'applicazione originale non si è dimostrata valida, oppure quando si è in procinto di cambiare radicalmente l'applicazione.
La riscrittura completa dell'applicazione ha il vantaggio che non dobbiamo portarci dietro scelte sbagliate fatte in passato.

.NET È APERTO?

Una delle domande più frequenti su .NET ed in particolare sul Framework è la sua apertura verso gli standard e la sua interoperabilità con altre piattaforme. La risposta non può che essere affermativa. .NET nasce basando gran parte della sua architettura su XML e sugli XML Web Services, permettendo l'interoperabilità dei servizi e delle applicazioni con altre realizzate su altre piattaforme.

Bisogna anche considerare che il linguaggio C#, e una buona parte del Framework necessario al suo funzionamento sono stati standardizzati dall'ECMA (l'associazione dei produttori europei di computer), lo stesso ente che aveva standardizzato JavaScript, ed ora sono stati sottoposti all'ISO per un'ulteriore standardizzazione.
A riprova dell'apertura delle specifiche, è in fase di realizzazione un'implementazione open source di .NET chiamata MONO che può girare su Windows, Linux e altre piattaforme. Microsoft stessa ha rilasciato una versione del Framework, di C#, di Jscript .NET che funziona sia su Windows, sia su FreeBSD.

TOOL DI SVILUPPO

Il .NET Framework ed i suoi compilatori sono completamente gratuiti, ma il modo migliore di apprezzare tutte le novità è quello di usare il Visual Studio .NET Questo tool integra centinaia di procedure automatizzate, ausili al programmatore, Wizard e quant'altro possa servire a rendere più produttivo lo sviluppo. Con una sola interfaccia utente si può programmare con Visual C++ .NET, Visual Basic.NET, Visual C# e con tutti gli altri linguaggi opzionali che si possono installare.
Sempre all'interno dello stesso tool si possono programmare nativamente le applicazioni Web in uno dei linguaggi supportati. L'anno prossimo dovrebbe uscire uno strumento di Borland che si annuncia come un valido concorrente dell'ambiente del Visual Studio.NET, e prevederà anche il supporto di Delphi nativo per .NET.
Se si devono solo sviluppare applicazioni ASP.NET è disponibile il tool gratuito ASP.NET Web Matrix Project, che permette di realizzare pagine ASP.NET in maniera molto semplice ed intuitiva.
Il supporto ASP.NET è presente anche in Macromedia DreamWeaver MX.
Esistono poi alcuni tool gratuiti che permettono di scrivere codice .NET senza dover acquistare un tool dedicato.

UNO SGUARDO AL FUTURO

Come tutti i prodotti software .NET non è nato perfetto e nemmeno completo di tutte le possibili funzionalità. Alcune di queste funzionalità sono già state aggiunte dopo il rilascio, come il linguaggio Visual J# presentato a luglio, il Mobile Internet Toolkit, e le estensioni del Framework (attualmente in beta) per dispositivi Smart (basati su Windows CE) come i Pocket PC. Contestualmente al rilascio (previsto nei prossimi mesi) del Windows .NET Server 2003, ci sarà l'uscita ufficiale del .NET Framework 1.1 e del Visual Studio.NET 7.1 (chiamato in codice Everett). Attualmente entrambi i prodotti sono in fase di beta testing, ma il loro rilascio non dovrebbe tardare. Il successivo rilascio dovrebbe poi coincidere con l'uscita del nuovo SQL Server (nome in codice Yukon) ma ci vorrà almeno un altro anno.
Il nuovo SQL Server integrerà direttamente il Framework al suo interno, e questo renderà possibile la realizzazione di stored procedure in linguaggi .NET oltre che in T-SQL.
Non c'è da preoccuparsi della compatibilità tra le varie versioni del Framework, in quanto queste possono coesistere tranquillamente sulla stessa macchina.

CONCLUSIONI

Possiamo ora rispondere in maniera completa ed esauriente alla domanda posta nel titolo: Microsoft .NET, conviene davvero?
Secondo noi si…